به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

هوش مصنوعی قابل اعتماد

تمرین سوم

نام و نام خانوادگی	نیما مدیرکیاسرایی
شماره دانشجویی	۸۱۰۱۰۲۳۳۹
تاریخ ارسال گزارش	۱۴۰۳/۰۳/۱۹

فهرست

سؤال اول

زیر بخش اول

$$P_X(Y = 1|A = O) = \sum_{S \in \{L,R\}} P_X(Y = 1|A = O, S) \times P_X(A = O|S)$$

$$= P_X(Y = 1|A = O, S = L) \times P_X(A = O, S = L) + P_X(Y = 1|A = O, S = R) \times P_X(A = O, S = R)$$

$$= \big(0.69 \times (1 - 0.77)\big) + \big(0.87 \times (1 - 0.24)\big) = \mathbf{0.8199}$$

$$P_X(Y = 1|A = N) = \sum_{S \in \{L,R\}} P_X(Y = 1|A = N, S) \times P_X(A = N|S)$$

$$= P_X(Y = 1|A = N\ S = L) \times P_X(A = N, S = L) + P_X(Y = 1|A = N, S = R) \times P_X(A = N, S = R)$$

$$= (0.73 \times 0.77) + (0.93 \times 0.24) = \mathbf{0.7853}$$

زیر بخش دوم

به دلیل اینکه متغیر A، Cause متغیر Y می باشد و در بخش شرط احتمال آماده است، بنابراین وجود do-notation تاثیری در جواب ندارد و جواب مانند زیر بخش قبل باقی می ماند:

$$P_X\big(Y = 1\big|do(A = O)\big) = P_X(Y = 1|A = O) = \mathbf{0.8199}$$

$$P_X(Y = 1|do(A = N)) = P_X(Y = 1|A = N) = \mathbf{0.7853}$$

سؤال دوم

A = [75000, 25000]

$Classifier: x_2 = -\frac{1}{5}x_1 + 45000 \qquad SCM: x_2 = \frac{3}{10}x_1 + u2$

$25000 = \left(75000 \times \frac{3}{10}\right) + u2 \rightarrow u_2 = 2500 \rightarrow SCM: x_2 = \frac{3}{10}x_1 + 2500$

$Cost = \sum_i \frac{|\delta_i|}{R_i} \qquad R_1 = 10^4 \times 10 = 10^5 \qquad R_2 = \left(\frac{3}{10} \times R_1\right) + 2500 = 32500$

تقاطع دو خط $Classifier$ و SCM : $\frac{3}{10}x_1 + 2500 = -\frac{1}{5}x_1 + 45000 \rightarrow \begin{cases} x_1 = 85000 \\ x_2 = 28000 \end{cases}$

$\left(-\frac{1}{5} \times 75000\right) + 45000 = 30000 \rightarrow \delta_2^* = 30000 - 25000 = 5000$

$Cost_2 = \frac{5000}{32500} = 0.15 \qquad Cost_1 = \frac{85000 - 75000}{10^5} = 0.1 \rightarrow$ تغییر دادن x_1 بهینه تر است

$\boldsymbol{A^* = do(X_1 := x_1^F + 10000) \rightarrow x^{SCF} = [85000, 28000]}$

A = [70000, 23800]

$Classifier: x_2 = -\frac{1}{5}x_1 + 45000 \qquad SCM: x_2 = \frac{3}{10}x_1 + u2$

$23800 = \left(70000 \times \frac{3}{10}\right) + u2 \rightarrow u_2 = 2800 \rightarrow SCM: x_2 = \frac{3}{10}x_1 + 2800$

$Cost = \sum_i \frac{|\delta_i|}{R_i} \qquad R_1 = 10^4 \times 10 = 10^5 \qquad R_2 = \left(\frac{3}{10} \times R_1\right) + 2800 = 32800$

تقاطع دو خط $Classifier$ و SCM : $\frac{3}{10}x_1 + 2800 = -\frac{1}{5}x_1 + 45000 \rightarrow \begin{cases} x_1 = 84400 \\ x_2 = 28120 \end{cases}$

$\left(-\frac{1}{5} \times 70000\right) + 45000 = 31000 \rightarrow \delta_2^* = 31000 - 23800 = 7200$

$Cost_2 = \frac{7200}{32800} = 0.219 \qquad Cost_1 = \frac{84400 - 70000}{10^5} = 0.144$

$\rightarrow$ تغییر دادن x_1 بهینه تر است

$\boldsymbol{A^* = do(X_1 := x_1^F + 14400) \rightarrow x^{SCF} = [84400, 28120]}$

سؤال سوم

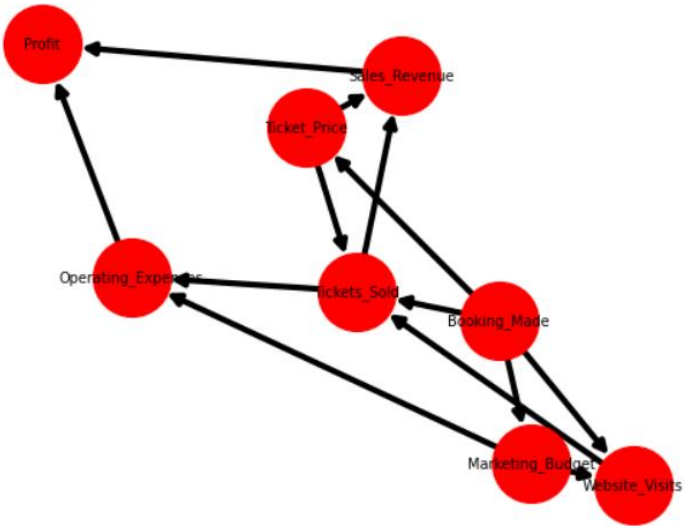
زیر بخش اول

در ابتدا نگاهی به دیتاست می اندازیم تا دید بهتری نسبت به دیتاها و ویژگی هایی که با آن سر و کار داریم داشته باشیم.

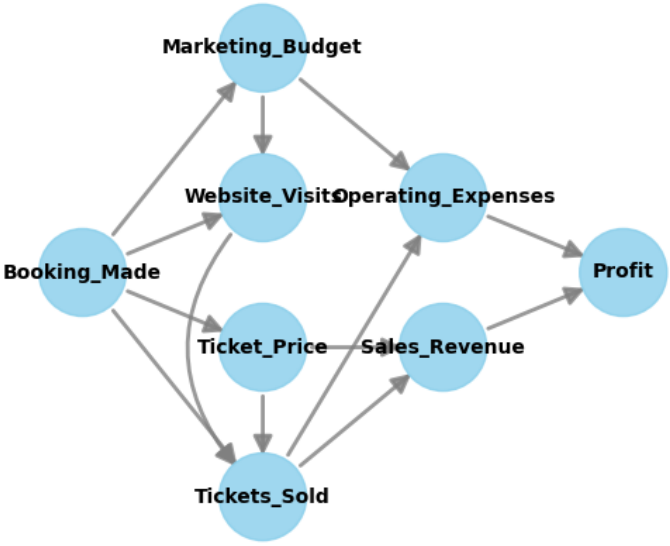
Date	Booking_Made	Marketing_Budget	Website_Visits	Ticket_Price	Tickets_Sold	Sales_Revenue	Operating_Expenses	Profit
2021-01-01	False	1217.741919	11789	1000.0	2389	2389000.0	1.695727e+06	693273.009394
2021-01-02	False	1462.814357	11778	1000.0	2381	2381000.0	1.691967e+06	689032.764060
2021-01-03	False	1498.712891	11804	1000.0	2369	2369000.0	1.686001e+06	682999.025547
2021-01-04	False	1243.245804	11809	1000.0	2371	2371000.0	1.686766e+06	684234.448174
2021-01-05	False	1307.334457	11444	1000.0	2334	2334000.0	1.668317e+06	665683.266002
...	...	...	...	...	...	...	...	...
2021-12-27	False	1491.918477	11842	1000.0	2297	2297000.0	1.650014e+06	646985.859228
2021-12-28	False	1457.854575	11767	1000.0	2328	2328000.0	1.665461e+06	662538.676222
2021-12-29	False	1318.752677	11745	1000.0	2336	2336000.0	1.669319e+06	666680.987999
2021-12-30	False	1222.097942	11501	1000.0	2361	2361000.0	1.681725e+06	679274.936009
2021-12-31	False	1279.409023	11682	1000.0	2380	2380000.0	1.691290e+06	688709.810588

شکل ۱- دیتاست استفاده شده در سوال ۳

در ادامه با استفاده از کتابخانه های network و dowhy به دو روش، گراف علی بین ویژگی ها را رسم می کنیم.



شکل ۲- گراف رسم شده با استفاده از networkx



شکل ۳- گراف رسم شده با استفاده از dowhy

زیر بخش دوم

در این قسمت یک مدل علی ساختاری (SCM) بر اساس گراف رسم شده در قسمت قبل با استفاده از کتابخانه dowhy پیاده سازی می کنیم. با استفاده از این کتابخانه می توان از رگرسیون خطی برای مدل سازی هر متغیر از متغیرهای والد آن استفاده کرد و همچنین بهترین توزیع احتمالی نویز برای داده ها را قرار داد تا یک SCM خوب را داشته باشیم و در واقع تمامی این کارها را خود کتابخانه dowhy به طور اتوماتیک انجام می دهد.

```
# Create the structural causal model object
scm = gcm.StructuralCausalModel(G)

# Automatically assign generative models to each node based on the given data
auto_assignment_summary = gcm.auto.assign_causal_mechanisms(scm, df, override_models=True, quality=gcm.auto.AssignmentQuality.GOOD)
```

شکل ۴- پیاده سازی مدل SCM

خروجی auto_assignment_summary به صورت خلاصه:

گره Booking_Made

گره Booking_Made یک گره ریشه است. بنابراین، تخصیص 'توزیع تجربی' به گره نماینده توزیع حاشیه‌ای.

گره Marketing_Budget

گره Marketing_Budget یک گره غیر ریشه با داده‌های پیوسته است. تخصیص 'مدل نویز افزودنی با استفاده از رگرسیون خطی' به گره.

این نشان‌دهنده رابطه علّی به صورت Marketing_Budget := f(Booking_Made) + N است.

برای انتخاب مدل، مدل‌های زیر بر اساس معیار خطای میانگین مربعات (MSE) ارزیابی شدند:

رگرسیون خطی: ۱۵۰۰۴.۹۵۷۷۸۰۶۸۳۷۶۸

Pipeline(مراحل=[('polynomialfeatures', PolynomialFeatures(include_bias=False)),

۱۵۰۳۱.۸۹۴۶۰۸۳۹۴۵۲۵('linearregression', LinearRegression)]):

HistGradientBoostingRegressor: ۸۸۶۷۷.۷۰۰۷۶۶۷۶۱۶۹

گره Ticket_Price

گره Ticket_Price یک گره غیر ریشه با داده‌های پیوسته است. تخصیص 'مدل نویز افزودنی با استفاده از Pipeline' به گره.

این نشان‌دهنده رابطه علّی به صورت Ticket_Price := f(Booking_Made) + N است.

برای انتخاب مدل، مدل‌های زیر بر اساس معیار خطای میانگین مربعات (MSE) ارزیابی شدند:

رگرسیون خطی: ۱۴۵.۰۱۷۲۵۱۹۹۰۸۶۰۵۶

Pipeline(مراحل=[('polynomialfeatures', PolynomialFeatures(include_bias=False)),

('linearregression', LinearRegression)]): ۱۴۴.۹۶۳۷۱۳۴۳۰۳۲۸۰۳

رگرسیون خطی: ۱۴۵.۰۱۷۲۵۱۹۹۰۸۶۰۵۶

HistGradientBoostingRegressor: ۴۵۲.۲۲۹۴۵۱۲۵۲۰۵۴۰۵

گره Website_Visits

گره Website_Visits یک گره غیر ریشه با داده‌های گسسته است. تخصیص 'مدل نویز افزودنی گسسته با استفاده از رگرسیون خطی' به گره.

این نشان‌دهنده رابطه علّی به صورت Website_visits:= f(Booking_Made, Marketing_budget) + N است.

برای انتخاب مدل، مدل‌های زیر بر اساس معیار خطای میانگین مربعات (MSE) ارزیابی شدند:

رگرسیون خطی: ۱۴۳۲۲۲.۰۹۰۲۷۹۱۶۷۰۵

Pipeline(مراحل=[('polynomialfeatures', PolynomialFeatures(include_bias=False)),

('linearregression', LinearRegression)]): ۱۷۵۵۹۳.۵۰۱۵۱۷۶۱۱۹۶

HistGradientBoostingRegressor: ۱۳۷۴۷۴۶.۶۲۶۸۴۲۰۹۷

گره Tickets_Sold

گره Tickets_Sold یک گره غیر ریشه با داده‌های گسسته است. تخصیص 'مدل نویز افزودنی گسسته با استفاده از رگرسیون خطی' به گره.

این نشان دهنده رابطه علّی به صورت Tickets_Sold:= f(Booking_Made, Ticket_Price, Website_Visits) است.

رگرسیون خطی: ۱۲۴۱۳.۴۱۶۸۸۵۰۴۹۹۲۶

Pipeline(مراحل=[('polynomialfeatures', PolynomialFeatures(include_bias=False)),

('linearregression', LinearRegression)]): ۲۱۳۲۷.۵۹۵۹۸۳۲۶۰۲۶

HistGradientBoostingRegressor: ۲۲۷۴۲۱.۶۱۷۸۶۹۴۱۶۳

گره Sales_Revenue

گره Sales_Revenue یک گره غیر ریشه با داده‌های پیوسته است. تخصیص 'مدل نویز افزودنی با استفاده از Pipeline' به گره.

این نشان‌دهنده رابطه علّی به صورت Sales_Revenue := f(Ticket_Price,Tickets_Sold) + N است.

برای انتخاب مدل، مدل‌های زیر بر اساس معیار خطای میانگین مربعات (MSE) ارزیابی شدند:

Pipeline(مراحل=[('polynomialfeatures', PolynomialFeatures(include_bias=False)),

('linearregression', LinearRegression)]): e-195.۱۳۸۸۲۱۲۵۵۸۹۰۱۹۹

رگرسیون خطی: ۱۷۵۶۶۲۶۹۵.۹۲۹۲۷۹۳۶

HistGradientBoostingRegressor: ۱۱۰۸۲۶۹۴۴۵۹۹.۸۵۸۷۳

گره Operating_Expenses

گره Operating_Expenses یک گره غیر ریشه با داده‌های پیوسته است. تخصیص 'مدل نویز افزودنی با استفاده از رگرسیون خطی' به گره.

این نشان‌دهنده رابطه علّی به صورت Operating_Expenses := f(Marketing_Budget,Tickets_Sold) + N است.

برای انتخاب مدل، مدل‌های زیر بر اساس معیار خطای میانگین مربعات (MSE) ارزیابی شدند:

رگرسیون خطی: ۳۷.۴۴۹۲۵۵۲۸۷۲۱

Pipeline(مراحل=[('polynomialfeatures', PolynomialFeatures(include_bias=False)),

('linearregression', LinearRegression)]): ۴۰.۰۸۹۸۷۸۲۳۶۸۵۲۶۴

HistGradientBoostingRegressor: ۱۲۰۵۳۹۷۲۱۶۵.۴۲۴۰۷۶

گره Profit

گره Profit یک گره غیر ریشه با داده‌های پیوسته است. تخصیص 'مدل نویز افزودنی با استفاده از رگرسیون خطی' به گره.

این نشان‌دهنده رابطه علّی به صورت Profit := f(Operating_Expenses,Sales_Revenue) + N است.

برای انتخاب مدل، مدل‌های زیر بر اساس معیار خطای میانگین مربعات (MSE) ارزیابی شدند:

رگرسیون خطی: ۶.۹۱۲۹۰۲۷۵۵۹۳۶۶۸۶ e-19

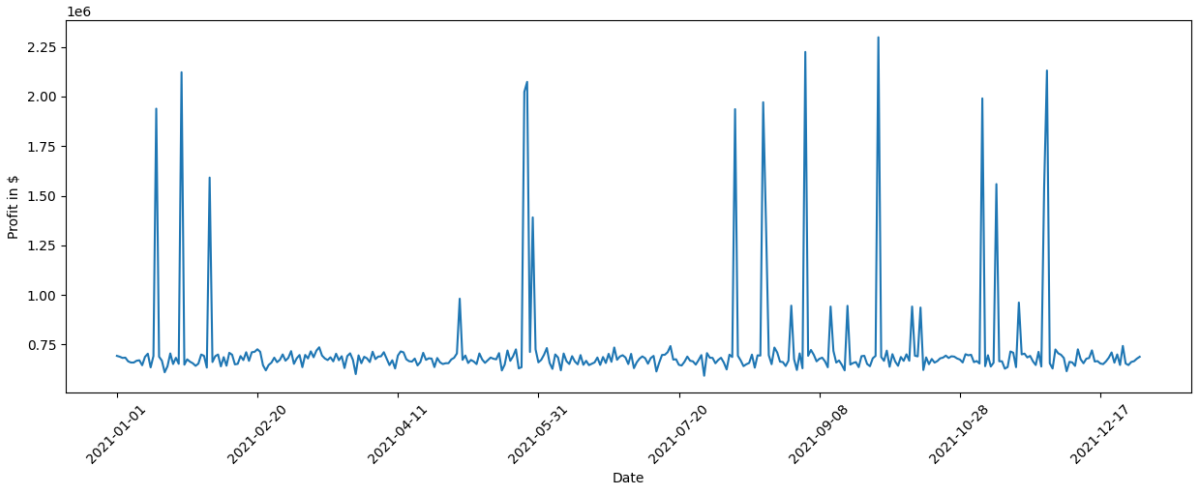
Pipeline(مراحل=[('polynomialfeatures', PolynomialFeatures(include_bias=False)),

('linearregression', LinearRegression)]): ۱.۵۰۲۶۸۹۹۵۲۷۲۰۷۴۳۵ e-6

HistGradientBoostingRegressor: ۱۹۹۷۲۳۰۳۲۸۶.۹۴۳۵۸

زیر بخش سوم

ابتدا مقادیر سود در گذر زمان در سال ۲۰۲۱ را رسم می کنیم تا توزیع آن را ببینیم:



شکل ۵- نمودار تغییر سود

همانطور که می بینیم در بعضی از مواقع، پیک های شدید و تندی در نمودار دیده می شود که قطعا باعث افزایش شدید واریانس سود می شود که در ادامه آن را بدست می آوریم.

```
df['Profit'].var()

62277529936.7847
```

شکل ۶- واریانس سود

همانطور که انتظار داشتیم، واریانس سود به شدت بالاست.

با توجه به گراف علی رسم شده در بخش اول می توانیم ببینیم که هزینه های عملیاتی و درآمد تاثیر مستقیمی در سود دارند و در این قسمت می خواهیم سهم یا درصد تاثیر هر یک از این دو ویژگی را روی مقدار سود بدست آوریم. با استفاده از متد arrow_strengths می توانیم این کار را انجام دهیم تا نتیجه را مشاهده کنیم:

```
('Operating_Expenses', 'Profit'): 25.127083749876316
('Sales_Revenue', 'Profit'): 74.87291625012368
```

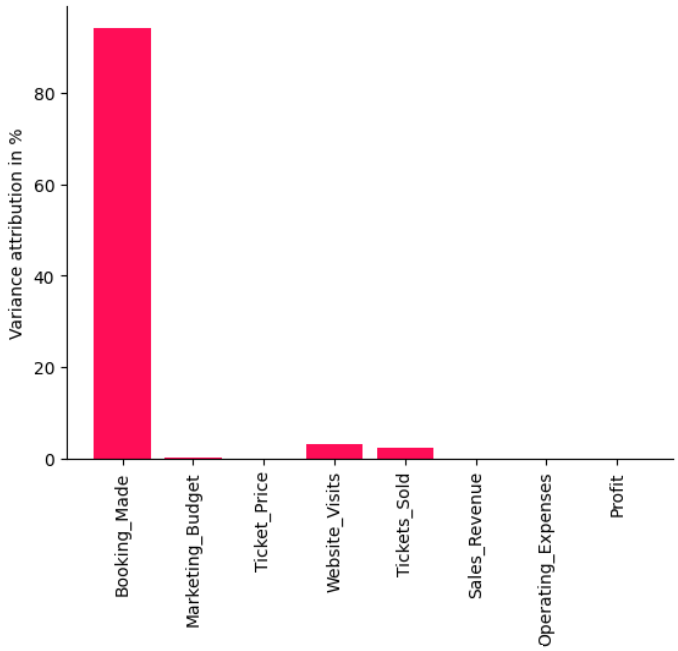
شکل ۷- سهم هر یک از گره های والد سود در میزان سود

از آنجایی که خود سود تنها تفاوت بین درآمد (Revenue) و هزینه‌های عملیاتی (Operational Cost) است، انتظار نداریم که عوامل بیشتری واریانس را تحت تأثیر قرار دهند. همانطور که می‌بینیم، درآمد تاثیر بیشتری نسبت به هزینه‌های عملیاتی دارد. این منطقی است زیرا درآمد معمولاً به دلیل وابستگی قوی‌تر به تعداد بلیت های فروخته شده، بیشتر از هزینه‌های عملیاتی تغییر می‌کند.

اما سوال اینکه کدام عامل در نهایت مسئول این واریانس بالا است، همچنان نامشخص است. به عنوان مثال، خود درآمد بر اساس تعداد بلیت های فروخته شده و قیمت واحد است. این سوال را در بخش بعدی جواب خواهیم داد.

زیر بخش چهارم

در این بخش تاثیر هر یک از ویژگی ها را در میزان واریانس سود را بدست می آوریم:



شکل ۸- تاثیر هر کدام از ویژگی ها در سود

در تصویر بالا می توانیم ببینیم که بیشترین سهم در واریانس سود را ویژگی Booking_Made دارد که تقریبا ۹۵ درصد سهم را دارد که منطقی هم بنظر میرسد. اینکه رزرو بلیت در روز های خاصی مانند تعطیلات یا مثلا Black Fridays صورت گرفته است تاثیر خیلی زیاد در سود دارد. بعد از آن ویژگی های Website_visits و Tickets_Sold بیشترین تاثیر در واریانس سود را دارند. در واقع می توان مقدار زیادی از واریانس سود را از مقدار Booking_Made توضیح داد (Explainability).

زیر بخش پنجم

در این بخش دیتای اولین روز سال بعد را بدست آورده ایم و می خواهیم آن را از لحاظ سود نسبت به روز اول سال گذشته مقایسه کنیم.

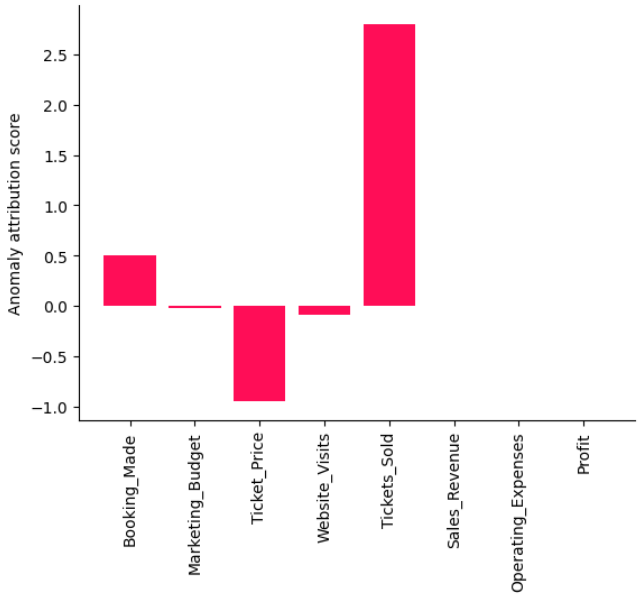
	Booking_Made	Marketing_Budget	Website_Visits	Ticket_Price	Tickets_Sold	Sales_Revenue	Operating_Expenses	Profit
0	True	2079.01	21110	700.47	7987	5594652.87	4495588.74	1099064.13

شکل ۹- دیتای اولین روز سال بعد

```
The profit increased by 0.5853265814580071%
```

شکل ۱۰- تغییر سود در اولین روز سال جدید نسبت به اولین روز سال قبل

می توانیم ببینیم که سود در ابتدای سال جدید نسبت به ابتدای سال گذشته تقریبا ۰.۵۸٪ افزایش داشته است. در ادامه با استفاده از متد attribute_anomalies می توانیم به ویژگی های مختلف برای سهم در افزایش سود امتیاز دهیم که در ادامه نتیجه آن را میبینیم.



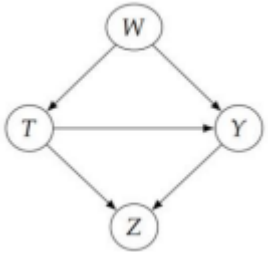
شکل ۱۱- تاثیر هر کدام از ویژگی ها در افزایش سود

در شکل بالا می توانیم ببینیم که ویژگی ای که بیشترین تاثیر را در افزایش سود نسبت به سال گذشته را داشته است Tickets_Sold می باشد و بعد از آن ویژگی Booking_Made اثر گذار ترین در جهت مثبت (کمک به افزایش سود) هستند. بر خلاف آن ها، ویژگی های Ticket_Price در درجه اول و Website_Visits امتیاز منفی دارند و کمک به کاهش سود می کنند اما همانطور که معلوم است زور

ویژگی های در جهت مثبت چربیده است و سود افزایش داشته است. به عنوان مثال مقدار بلیت های فروخته شده روزانه در سال گذشته تقریبا حوالی ۲۰۰۰ تا بوده است اما در اولین روز سال جدید ۷۹۸۷ عدد بوده است که قطعا تاثیر زیادی در افزایش سود داشته است. اما برعکس قیمت بلیت که سال قبل ۱۰۰۰ دلار بوده است در اولین روز سال جدید به ۷۰۰.۴۷ دلار کاهش یافته است که کمک به کاهش سود می کند. بنابراین نتیجه نهایی (افزایش سود) منطقی به نظر می رسد.

سؤال چهارم

در این سوال با استفاده از دیتاست health و گراف علی رسم شده در شکل زیر می خواهیم اثر Insulin (گره T) بر blood_glucose (گره Y) را بررسی کنیم.



شکل۱۲- گراف مربوط به دیتاست health

در ادامه مقادیر خواسته شده در صورت سوال را محاسبه می کنیم:

۱) برای یافتن مقدار $E_{W,Z}E_Y[y \mid t, W, Z]$ باید بر اساس ویژگی های Age، insulin_level و blood_pressure و خروجی blood_glucose یک مدل LinearRegression تخمین بزنیم و در نهایت ضریب متغیر insulin_level را گزارش می کنیم.

```
0.8609542611622993
```

۲) برای یافتن مقدار $E_W E_Y[y \mid t, W]$ باید بر اساس ویژگی های Age و insulin_level و خروجی blood_glucose یک مدل LinearRegression تخمین بزنیم و در نهایت ضریب متغیر insulin_level را گزارش می کنیم.

```
1.0507202300104845
```

۳) برای یافتن مقدار $E_Y[y \mid t]$ باید بر اساس ویژگی insulin_level و خروجی blood_glucose یک مدل LinearRegression تخمین بزنیم و در نهایت ضریب متغیر insulin_level را گزارش می کنیم.

```
3.6190643289837436
```

.

ابتدا باید فرضیات علّی خود را به صورت یک نمودار علّی بنویسیم. متغیرهای کمکی عبارتند از سن(W) و فشار خون(Z) . سن یک علت مشترک برای هر دو ویژگی فشار خون و سطح انسولین می باشد. در مقابل، مقدار بالای فشار خون توسط مقدار بالای گلوکز خون و سطح بالای ایجاد می‌شود. این بدان معناست که فشار خون یک هم‌پوشان (Collider) است.

چون Z یک هم‌پوشان است، شرطی‌سازی بر روی آن باعث القای بایاس می‌شود. حال که روابط علّی را با نمودار علّی مشخص کردیم، معیار backdoor adjustment به ما می‌گوید که فقط باید برای W تنظیم کنیم و برای Z تنظیم نکنیم. به طور دقیق‌تر، از تنظیم درب پشتی (قضیه ۴.۲) استفاده خواهیم کرد تا برآوردگر آماری خود را به شکل زیر تغییر دهیم:

$$E_W E_Y[y \mid t, W]$$

ما به سادگی متغیر هم پوشان Z را از متغیرهای تنظیم پذیر حذف کردیم. ما انتظار بیرونی بر روی W را با میانگین تجربی بر روی W جایگزین می‌کنیم و انتظار شرطی $E_Y[y \mid t, W]$ را با یک مدل یادگیری ماشین (در این مورد، رگرسیون خطی) جایگزین می‌کنیم.

سؤال پنجم

در ابتدا نگاهی به دیتاست می اندازیم تا دید بهتری نسبت به دیتاها و ویژگی هایی که با آن سر و کار داریم داشته باشیم.

	age	insulin	blood_glucose	blood_pressure	category
0	73.820262	4.410849	152.069798	49.855261	1
1	67.000786	2.984810	136.302391	42.671658	0
2	69.893690	2.346063	143.984346	43.736376	1
3	76.204466	3.671327	156.454474	48.838422	1
4	74.337790	2.530366	151.154654	46.858840	1
...	...	...	...	...	...
4995	64.493128	1.773670	131.477051	40.424245	0
4996	68.733329	3.860877	143.359459	44.443191	1
4997	69.645909	4.386089	144.650306	45.092268	1
4998	66.147090	3.641918	135.537161	41.058984	1
4999	67.072029	5.024335	139.221231	43.449575	1

شکل ۱۳- دیتاست health

زیر بخش اول

در این بخش تابع process_health_data از فایل data_utils.py را طوری تغییر می دهیم تا خواسته های سوال بر آورده شود.

```
227     ##### complete the first part ######
228
229     actionable = [1, 2]
230
231     ##### end of first part ######
232
233
234
235     ##### complete the second part ######
236
237     feature_limits = np.array([[-1, 1]]).repeat(X_health.shape[1], axis=0) * 1e10
238     feature_limits[1][0] = min(X_health[:, 1])
239     feature_limits[1][1] = max(X_health[:, 1])
240     feature_limits[2][0] = min(X_health[:, 2])
241     feature_limits[2][1] = max(X_health[:, 2])
242     feature_limits[3][0] = min(X_health[:, 3])
243     feature_limits[3][1] = max(X_health[:, 3])
244
245
246
247     ##### end of the second part  ######
```

شکل ۱۴- کد مربوط به کامل کردن تابع process_health_data

زیر بخش دوم

در این قسمت فایل main.py را برای ۱۰ فرد ناسالم اجرا می کنیم و هزینه محاسبه شده را گزارش می کنیم.

```
E: 30 Acc: 0.8890 mcc: 0.7794
E: 31 Acc: 0.8900 mcc: 0.7827
E: 32 Acc: 0.8900 mcc: 0.7827
E: 33 Acc: 0.8910 mcc: 0.7840
E: 34 Acc: 0.8890 mcc: 0.7794
E: 35 Acc: 0.8910 mcc: 0.7845
E: 36 Acc: 0.8900 mcc: 0.7817
E: 37 Acc: 0.8910 mcc: 0.7845
E: 38 Acc: 0.8900 mcc: 0.7812
E: 39 Acc: 0.8900 mcc: 0.7812
E: 40 Acc: 0.8900 mcc: 0.7827
E: 41 Acc: 0.8900 mcc: 0.7812
E: 42 Acc: 0.8890 mcc: 0.7794
E: 43 Acc: 0.8890 mcc: 0.7794
E: 44 Acc: 0.8890 mcc: 0.7794
E: 45 Acc: 0.8880 mcc: 0.7776
E: 46 Acc: 0.8890 mcc: 0.7794
E: 47 Acc: 0.8890 mcc: 0.7794
E: 48 Acc: 0.8890 mcc: 0.7794
E: 49 Acc: 0.8900 mcc: 0.7812
```

شکل ۱۵- بخش آخر فاز آموزش مدل

```
Valid recourse: 1.000
Cost recourse: 1.132
```

شکل ۱۶- نتیجه فاز evaluation و مقدار هزینه محاسبه شده

می توانیم ببینیم که هزینه محاسبه شده برابر با ۱.۱۳۲ می باشد. تعریف هزینه یا Cost در این بخش نیز مانند تعریف Cost در حل سوال ۲ می باشد:

$$Cost = \sum_i \frac{|\delta_i|}{R_i}$$

در رابطه بالا δ مقدار intervention برای هر متغیر و R مقدار ماکزیمم هر متغیر می باشد. در واقع برای محاسبه هزینه، مقدار تغییری که در هر متغیر ایجاد شده است را تقسیم بر مقدار ماکزیمم آن متغیر می کنیم و این کار را برای همه ی متغیرها انجام می دهیم و این مقادیر را با هم جمع می کنیم تا Cost بدست آید. در واقع هزینه گزارش شده میانگین هزینه بدست آوردن بهترین و به صرفه ترین مداخله برای تغییر فرد بیمار به سالم برای ۱۰ نفر می باشد.

زیر بخش سوم

در این بخش، کلاس HEALTH_SCM در فایل scm.py را طوری کامل می کنیم تا خواسته های سوال برآورده شود.

```
675         ##### complete the first part ######
676
677         self.actionable = [1, 2]
678         self.soft_interv = [True, False, False, True]
679
680         ##### end of first part ######
```

شکل ۱۷- کد مربوط به کامل کردن بخش اول کلاس HEALTH_SCM

زیر بخش چهارم

در این بخش تابع get_jacobian در فایل scm.py را طوری تکمیل می کنیم که ژاکوبین SCM را به عنوان خروجی برگرداند.

این ماتریس یک ماتریس پایین مثلثی می باشد که مقادیر قطر اصلی آن برابر ۱ می باشد. برای پر کردن بقیه عناصر آن، از این الگوریتم استفاده می کنیم: برای عنصر (i, j) تمام مسیر های از گره j به گره i را پیدا می کنیم و وزن های مسیر های مختلف را با هم جمع می کنیم. در ضمن اگر در طی کردن هر مسیری از گره های دیگر رد شدیم، باید وزن های دیده شده را در یکدیگر ضرب کرده و در نهایت مقادیر را جمع کنیم.

```
689         ##### complete the second part ######
690
691         Jacobi = np.array([[1, 0, 0, 0],
692                            [w21, 1, 0, 0],
693                            [w31 + w21*w32, w32, 1, 0],
694                            [w21*w42 + w31*w43 + w21*w32*w43, w42 + w43*w32, w43, 1]])
695         ##### end of the second part  ######
696
697         return Jacobi
```

شکل ۱۸- کد مربوط به کامل کردن تابع get_jacobian

زیر بخش پنجم

در این قسمت کامنت های داخل تابع get_scm در فایل utils.py را حذف می کنیم تا SCM و روابط علی در محاسبات لحاظ شود.

```
Valid recourse: 1.000
Cost recourse: 0.898
```

شکل ۱۹- نتیجه فاز evaluation و مقدار هزینه محاسبه شده

در این قسمت نیز هزینه مانند بخش قبل طبق فرمول $Cost = \sum_i \frac{|\delta_i|}{R_i}$ محاسبه می شود و میانگین هزینه بدست آوردن بهترین و به صرفه ترین مداخله برای تغییر فرد بیمار به سالم برای ۱۰ نفر می باشد.

زیر بخش ششم

همانطور که مشاهده کردیم هزینه محاسبه شده در بخش B که CFE-based recourse می باشد، بیشتر از هزینه محاسبه شده در بخش E که بر اساس SCM-based recourse می باشد، است.

- A^{CFE} : Counter Factual Explanation-based action
- A^* : Minimal Intervention Solution (blue box above)

$$\mathbf{cost(A^*; x^F) \leq cost(A^{CFE}; x^F)}$$

دلیل آن هم این است که در روش SCM، روابط علی بین ویژگی ها یا متغیرها در نظر گرفته می شود اما در روش CFE این روابط در نظر گرفته نمی شود. وقتی این روابط در نظر گرفته می شود هزینه کمتری نیاز است تا متغیرها تغییر کنند و بهینه ترین مداخله برای تغییر وضعیت بیمار پیدا شود اما وقتی این روابط در نظر گرفته نشود احتمالا نیاز است تا متغیرهای بیشتری تغییر کند تا به بهینه ترین مداخله برسد و در نتیجه هزینه نیز افزایش پیدا می کند.

الگوریتم CFE-based recourse

هزینه محاسبه شده: ۰.۸۵۵۷۱۸۵۳

متغیرهایی که مداخله بر روی آنها انجام شده است: insulin level, blood glucose

تغییراتی که بر روی ویژگی ها انجام شده است: `[-5.84412780e-18  8.55718534e-01  8.43087993e-12 -5.76692160e-20]`

الگوریتم SCM-based recourse

هزینه محاسبه شده: ۰.۸۹۸

متغیرهایی که مداخله بر روی آنها انجام شده است: insulin level

تغییراتی که بر روی ویژگی ها انجام شده است: `[ 1.13671116e-17  6.79336851e-01  6.24595341e-18  1.19412174e-18]`

همانطور که می بینیم در حالت SCM هزینه کلی محاسبه شده، تعداد متغیرهای که در مداخله حضور دارند و intervention ها کمتر از حالت CFE می باشد.

سؤال ششم

زیر بخش اول

قضیه:

برای یک مدل ساختاری علی (SCM) با معادلات ساختاری خطی، اگر همه ویژگی‌ها قابل اقدام (actionable) باشند و یک $x^+ \in X$ منحصر به فرد وجود داشته باشد که به طور مقاوم طبقه‌بندی شده باشد به طوری که $h(\acute{x}) = 1 \; \forall \acute{x} \in B(\acute{x})$ در این صورت یک اقدام بازخوردی مقاوم در برابر حملات برای هر $x \in X$ وجود دارد.

قضیه:

برای یک طبقه‌بند خطی h و یک مدل ساختاری علی خطی M، تحت شرایط ملایم در وزن‌های طبقه‌بند که در پیوست C.4 توصیف شده‌اند، اگر ویژگی X_j وجود داشته باشد که قابل اقدام (actionable) و بدون محدودیت (unbounded) باشد، در این صورت یک اقدام بازخوردی مقاوم در برابر حملات برای هر $x \in X$ وجود دارد.

شرایط وزن های ملایم در پیوست C.4 مقاله:

ما استدلال می‌کنیم که شرط $\langle w, v \rangle \neq 0$ یک شرط ملایم بر وزن‌های طبقه‌بند است، دقیقاً در حالتی که وزن‌های طبقه‌بند به صورت خصمانه نسبت به مدل ساختاری علی (SCM) انتخاب نشده‌اند.

زیر بخش دوم

برای یک طبقه‌بند خطی $h(x) = \langle w, v \rangle \geq b$ و مدل ساختاری علی (SCM) خطی، نشان داده می شود که تولید بازخورد مقاوم برای طبقه‌بند h معادل تولید بازخورد استاندارد برای یک طبقه‌بند خطی اصلاح‌شده $\acute{h}(x) = \langle w, v \rangle \geq \acute{b}$ است که "آستانه پذیرش" آن به اندازه کافی افزایش یافته است.

Corollary 1

برای هر $x \in X$ منحصر به فرد اقدام بازخوردی مقاوم در برابر حملات با کمترین هزینه برای طبقه‌بند اصلی h معادل اقدام بازخوردی استاندارد با کمترین هزینه برای طبقه‌بند اصلاح‌شده $\acute{h}$ است.

بنابراین، در تنظیمات خطی، طبق Corollary 1 هر روشی برای تولید بازخورد استاندارد می‌تواند به راحتی برای تولید بازخورد مقاوم در برابر حملات استفاده شود، تنها با در نظر گرفتن طبقه‌بند اصلاح‌شده $\acute{h}$.در چنین مواردی، استحکام در برابر حملات می‌تواند به سادگی درون روش‌هایی که به دنبال ارتقاء دیگر معیارها مانند پشتیبانی داده بزرگ هستند جا داده شود.